Argonne NATIONAL LABORATORY

ALCF Early Science Program

ESP Kick-Off Workshop Project Plan Presentation

Ab-initio Reaction Calculations for Carbon-12

PI: Steven Pieper (Argonne)

Work with:

Ralph Butler (MSTU)

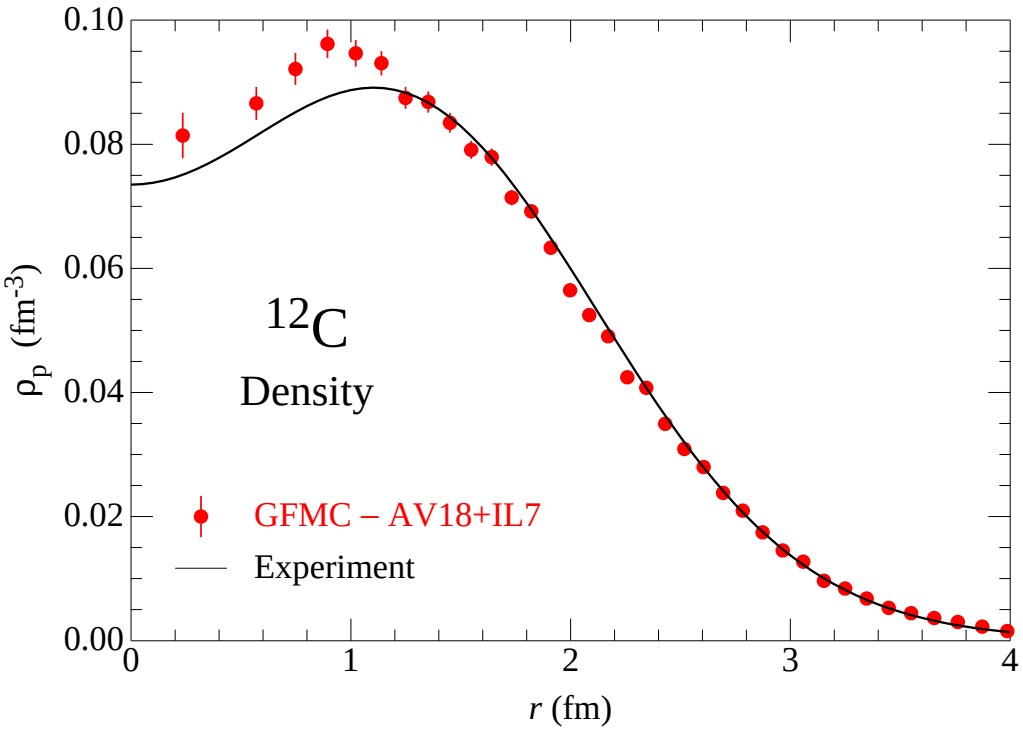
Joseph Carlson (LANL)

Rusty Lusk (Argonne)

Robert Wiringa (Argonne)

Presenter: Steven Pieper

October 18–19, 2010



PROJECT OVERVIEW

Calculate fundamental properties of ^{12}C nucleus by computing density matrix and response functions.

- Compute neutrino-nucleus reaction cross sections
  - Needed to analyze ongoing neutrino experiments
  - Only source of these cross sections, no experimental data available at present
- Compute electron-nucleus scattering cross sections
  - Needed to analyze quasi-elastic knock-out scattering experiments at JLab
- BG/Q enables proposed density-matrix and response calculations (too big for Intrepid)

Scientific Field: Nuclear Structure

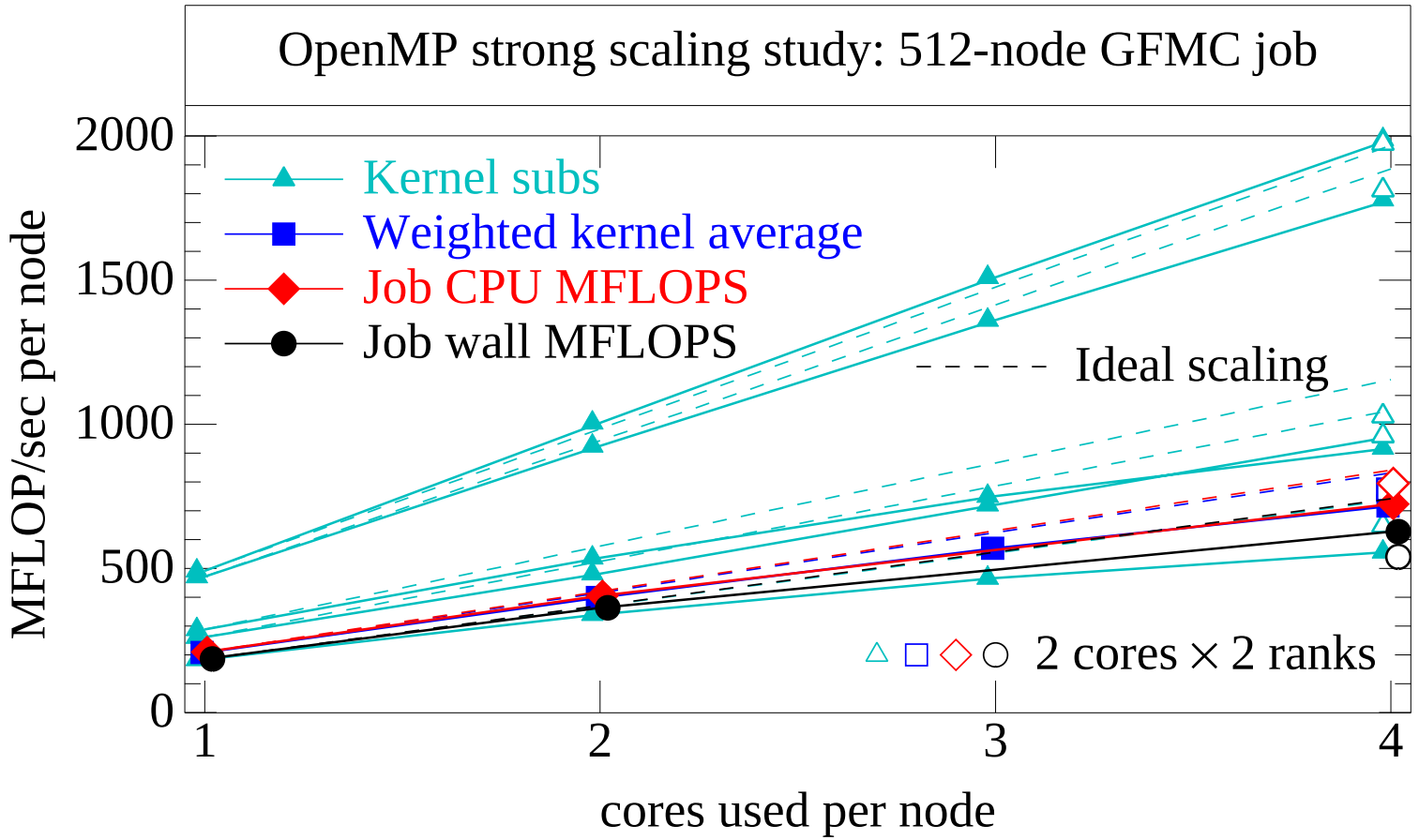
Codes: GFMC

Computational Approach, Numerical Methods

- Green's Function Monte Carlo (GFMC)
  - Starts with an approximate wave function (Ψ_T) and evolves it to the exact Ψ for the given nuclear interaction (Hamiltonian)
  - Evolution is done as a sequence of imaginary time steps
  - Each time step is a $3\times$(number-of-nucleons) integral
  - Result is a $\sim$70,000-dimensional integral done by Monte Carlo
- Dynamic load balancing
  - Monte Carlo integration done by branching random walk
  - Branching multiplies and destroys configurations – load fluctuates
  - Work for a single MC sample must be distributed to many nodes
- CPU time dominated by sparse matrix $\times$ vector operations
  - Kernel subtroutines based on explicit (complex) matrix structure
  - Recently improved by Vitali Morozov & James Osborn
  - OpenMP works well on BG/P for these

PARALLELISM AND EXISTING IMPLEMENTATION

- Automatic Dynamic Load Balancing (ADLB) & MPI used for inter-node communication
- OpenMP is used for the four cores on a BG/P node
- Current Performance/Scalability is quite good
- BG/P production jobs mostly done on 8 racks

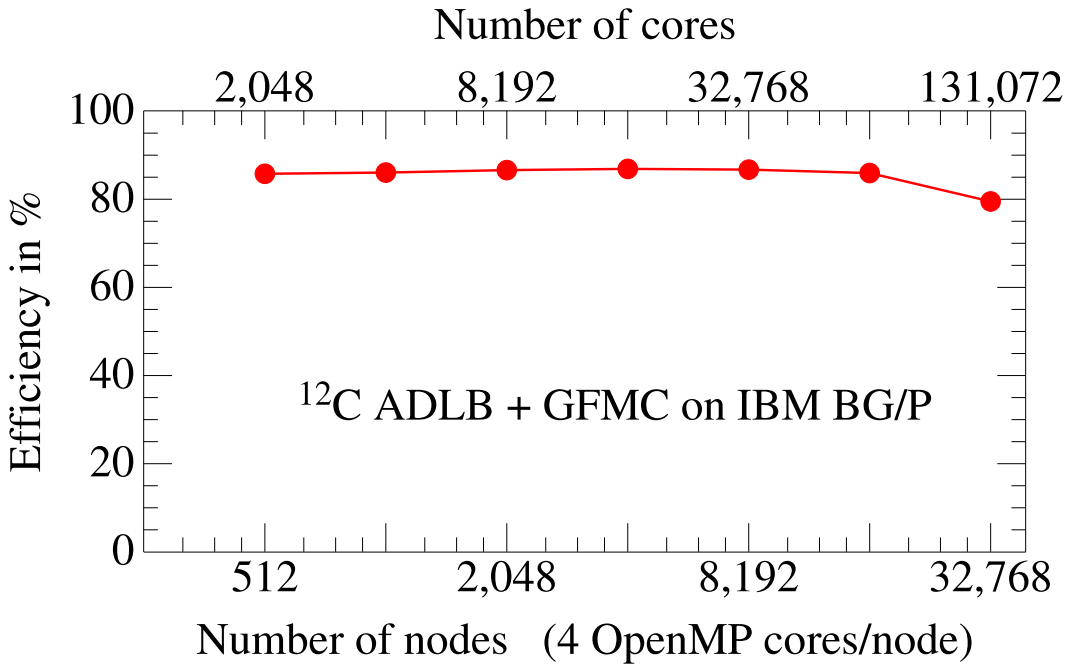


LIBRARY AND TOOL DEPENDENCIES

- Libraries: Automatic Dynamic Load Balancing (ADLB)
  - Developed by us as part of UNEDF SciDAC
  - Nodes put work packages into ADLB & request work from it
  - The GFMC code has been the principal test bed
  - Perfect scaling to 16,384 nodes, good to 32,768 nodes of BG/P
  - ADLB is a general purpose library; give it a try! –

    http://www.cs.mtsu.edu/∼rbutler/adlb

- Tools
  - gprof
  - jumpshot



ANTICIPATED MODIFICATIONS FOR BLUE GENE/Q

- We hope that OpenMP will scale well to 16 cores per node
  - Must use at least 2 cores/rank to get 2 Gbyte/rank
  - Should have no communication problems with one rank per node
  - Initial efforts will be concentrated on OpenMP
- Will also work on single core & thread performance improvement
- Current ADLB should work well; we are also designing a new version using MPI one-sided puts & gets
- We have to develop subroutines for the density matrix
- We have to develop Ψ_T for the excited states of ^{12}C

PLAN FOR NEXT SIX MONTHS EFFORT

- Help find and hire a project postdoc
- Continue work on one-sided version of ADLB
- Start work on density-matrix calculations (using small nuclei)
- Work on starting wave functions (Ψ_T) for ^{12}C states

AUTOMATIC DYNAMIC LOAD BALANCING – THE API

- Startup and termination
  - ADLB_Init(num_servers, am_server, app_communicator)
  - ADLB_Server()
  - ADLB_Set_No_More_Work()
  - ADLB_Finalize()
- Putting work or answers
  - ADLB_Begin_Batch_Put(common_buffer, length) – optional
  - ADLB_Put(type, priority, length, buffer, answer_destination)
  - ADLB_End_Batch_Put() – optional
- Getting work or answers
  - ADLB_Reserve(req_types, work_handle, length, type, priority, answer_destination)
  - or ADLB_Ireserve($\cdots$)
  - ADLB_Get_Reserved(work_handle, buffer)

ADLB – CURRENT GFMC IMPLEMENTATION

Old GFMC

Each slave gets several configurations

Slave

propagates configurations

(few w.f. evaluations)

replicates or kills configs (branching)

$\rightarrow$ periodic global redistribution

computes energies

(many w.f. evaluations)

Need $\sim$10 configs per slave

^{12}C will have only $\sim$10,000 configs.

Can't do on more than 2000 processors

Configurations cannot be unit of parallelization

With ADLB

A few "boss" slaves manage the propagation:

- Generate propagation work packages
  – Answers used to make 0,1,2,$\cdots$ new propagation packages (branching)
  – Number of prop. packages fluctuates
  – Global redistribution may be avoided
- Generate energy packages – No answers

When propagation done, become worker slaves

Most slaves ask ADLB for work packages:

- Propagation package
  – Makes w.f. and $3N$ potential packages
- Energy package
  – Makes many w.f. packages
  – Makes $3N$ potential packages
  – Result sent to Master for averaging
- Wave Function or $3N$ potential package
  – Result sent to requester

Wave function is parallelization unit

Can have many more nodes than configurations